ASSIGNMENT-1

Analyze a given business scenario and create an ER diagram that includes entities, relationships, attributes, and cardinality. Ensure that the diagram

Business Scenario:

A library has multiple branches, each of which contains many books. Each book is uniquely identified by its ISBN number, and each branch keeps track of the copies of books it holds. Members can borrow books from any branch, and each member can borrow multiple books. The library system needs to track information about books, branches, members, and borrowings.

Entities:

1. Book
2. Branch
3. Member
4. Borrowing

Relationships:

1. Book is available at Branch (Many-to-Many)
2. Member borrows Book (Many-to-Many)
3. Branch has copies of Book (One-to-Many)
4. Borrowing involves Member and Book (Many-to-Many)

Attributes:

1. Book:

- ISBN (Primary Key)
- Title
- Author
- Genre
- Publisher
- Publication Year

2. Branch:

- BranchID (Primary Key)
- BranchName
- Location
- Contact Number

3. Member:

- MemberID (Primary Key)
- Name
- Email
- Phone
- Address

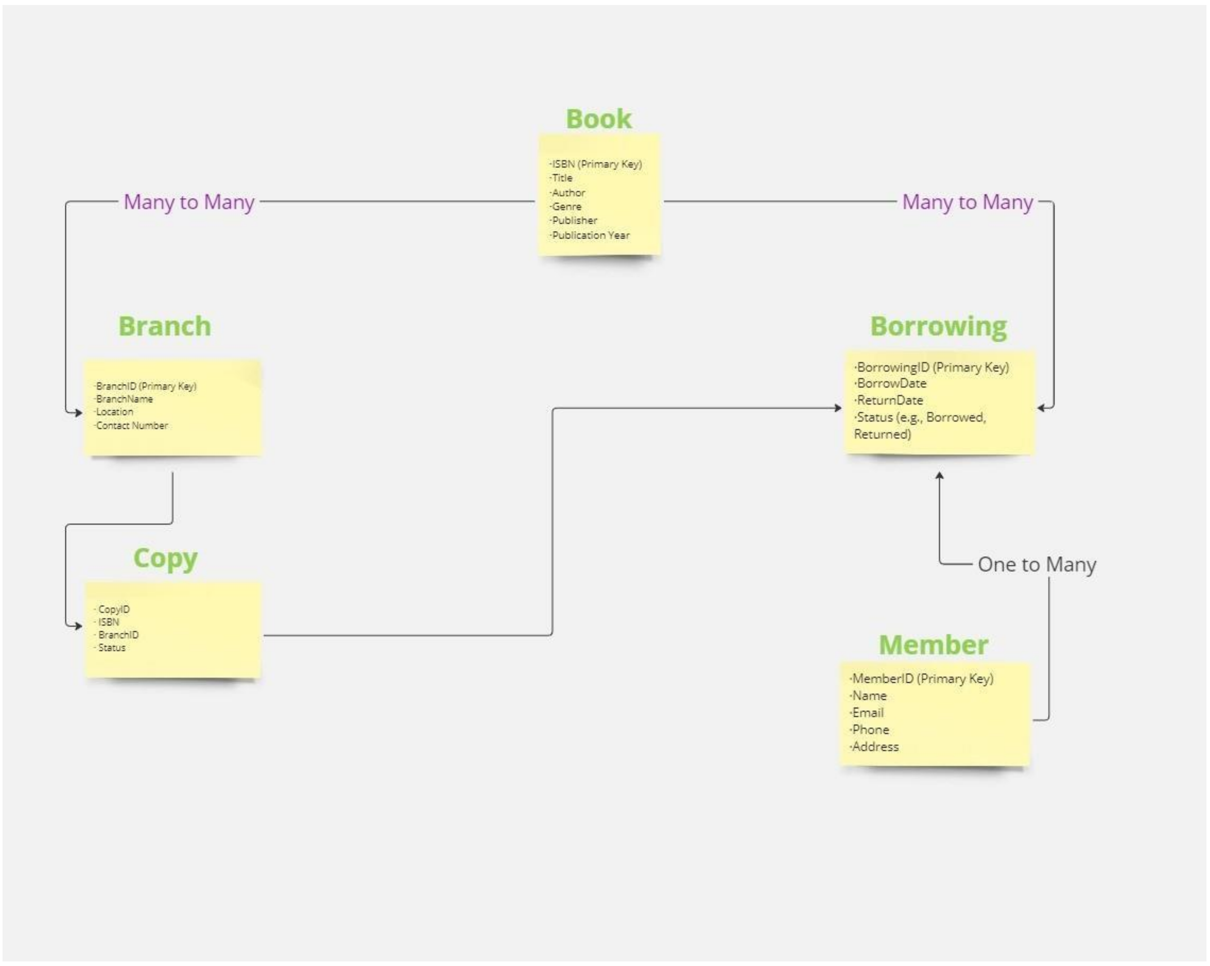
4. Borrowing:

- BorrowingID (Primary Key)
- BorrowDate
- ReturnDate
- Status (e.g., Borrowed, Returned) **Cardinality:**

- One Book can be available at Many Branches.

- One Branch can have Many Books.
- One Member can borrow Many Books.
- One Book can be borrowed by Many Members.
- One Borrowing involves One Member and One Book.

ER Diagram (Third Normal Form):



This ER diagram captures the entities (Book, Branch, Member, Borrowing) and their attributes, along with the relationships between them. The normalization up to the third normal form ensures that data redundancy is minimized and data integrity is maintained within the system.reflects proper

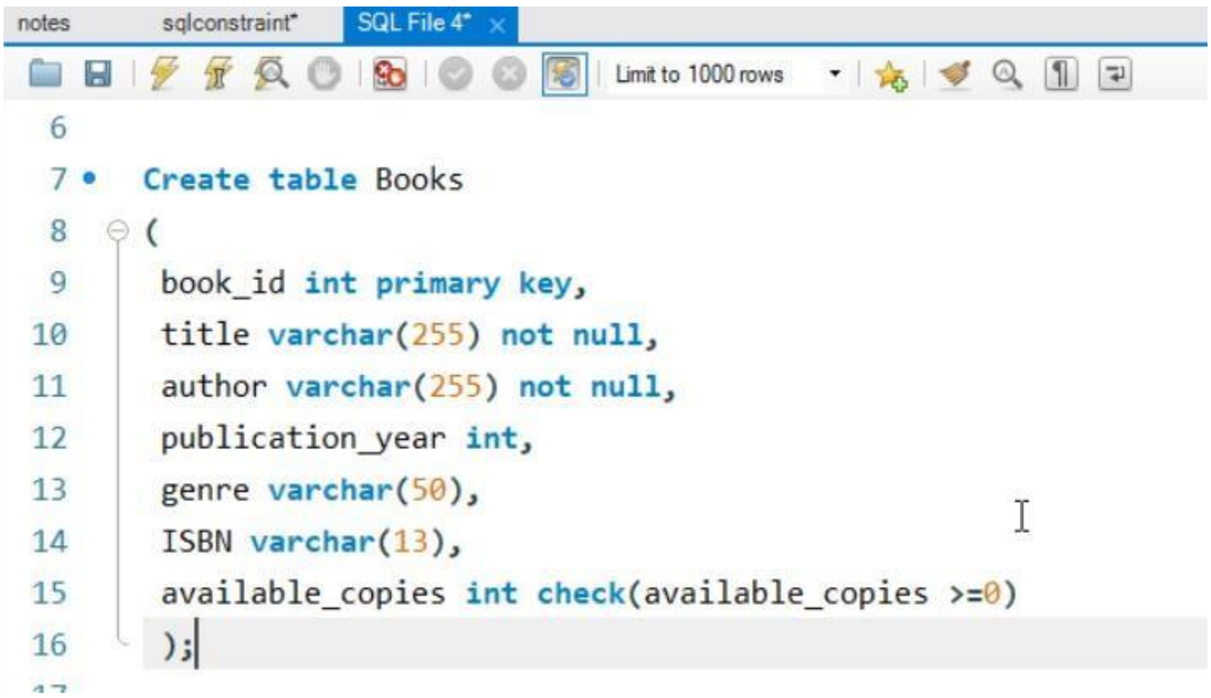
ASSIGNMENT-2

Design a database schema for a library system, including tables, fields, and constraints like NOT NULL, UNIQUE, and CHECK.

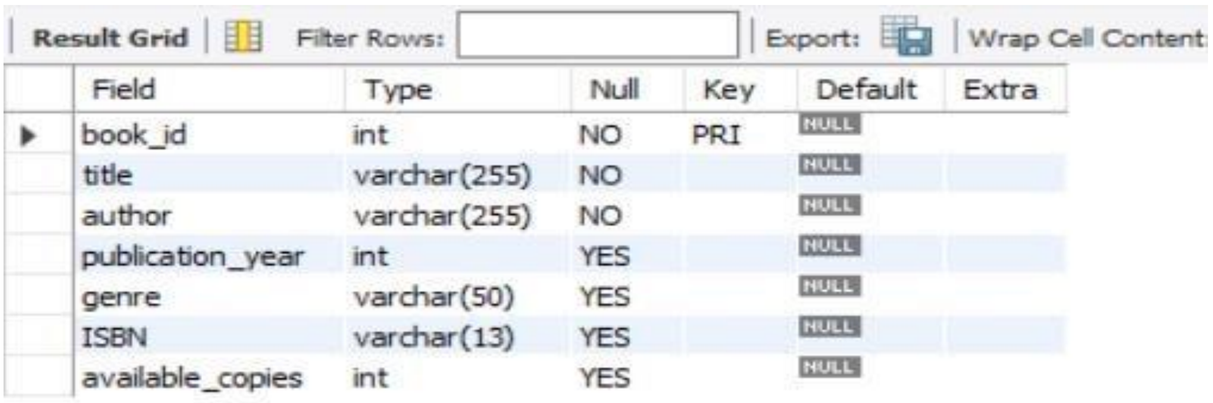
Include primary and foreign keys to establish relationships between tables.

1. **Books:** This table will store information about books in the library.

- **Fields:** book_id (Primary Key), title, author, publication_year, genre, ISBN (Unique), available_copies.



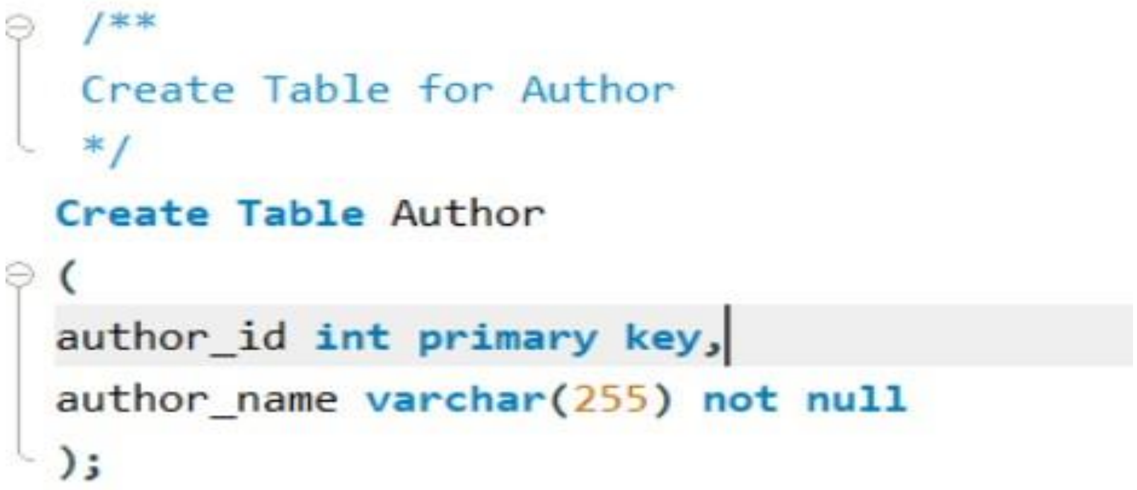
```sql

Create table Books
(
 book_id int primary key,
 title varchar(255) not null,
 author varchar(255) not null,
 publication_year int,
 genre varchar(50),
 ISBN varchar(13),
 available_copies int check(available_copies >=0)
 );
```



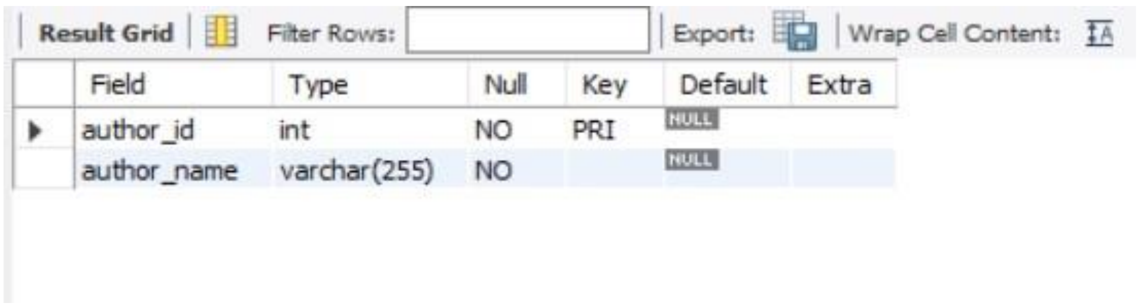
	Field	Type	Null	Key	Default	Extra
▶	book_id	int	NO	PRI	NULL	
	title	varchar(255)	NO		NULL	
	author	varchar(255)	NO		NULL	
	publication_year	int	YES		NULL	
	genre	varchar(50)	YES		NULL	
	ISBN	varchar(13)	YES		NULL	
	available_copies	int	YES		NULL	

2. **Authors:** This table will store information about authors.

• **Fields:** author_id (Primary Key), author_name



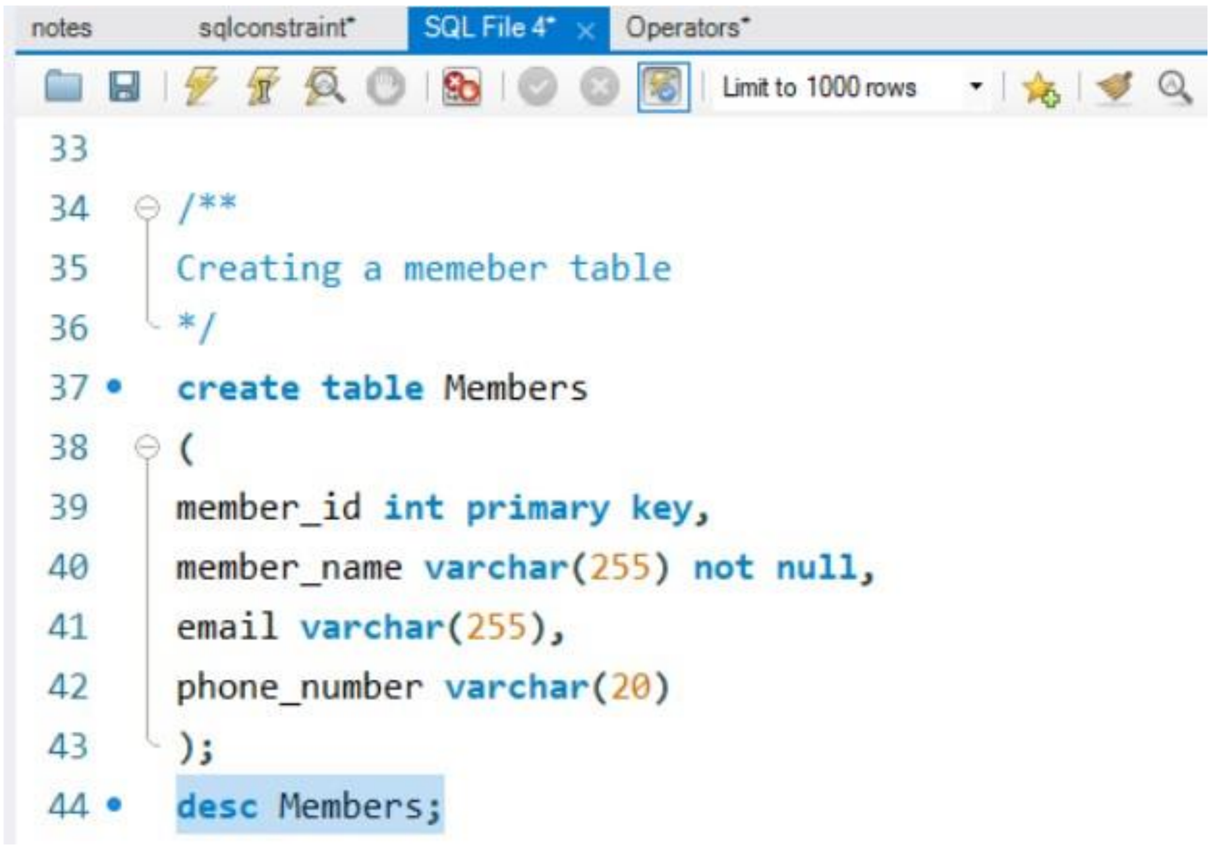
```
/**
Create Table for Author
*/
Create Table Author
(
author_id int primary key,
author_name varchar(255) not null
);
```


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

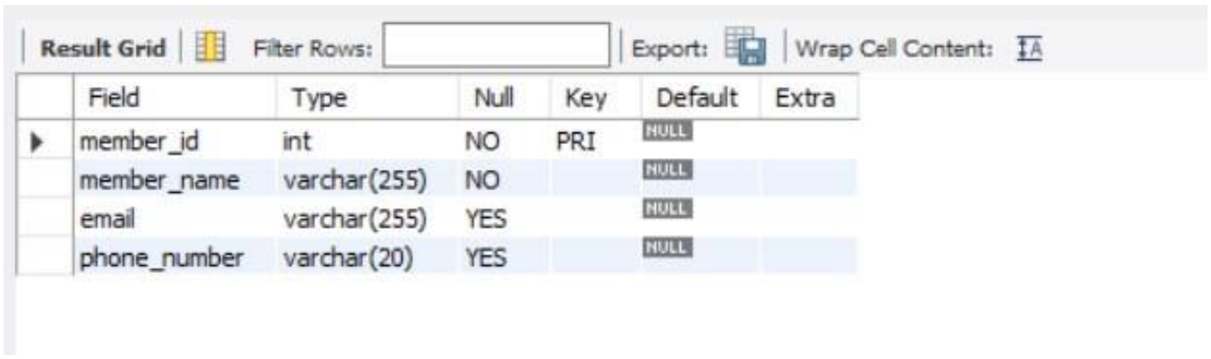
Field	Type	Null	Key	Default	Extra
author_id	int	NO	PRI	NULL	
author_name	varchar(255)	NO		NULL	

3. **Members:** This table will store information about library members.

- **Fields**: member_id (Primary Key), member_name, email, phone_number.



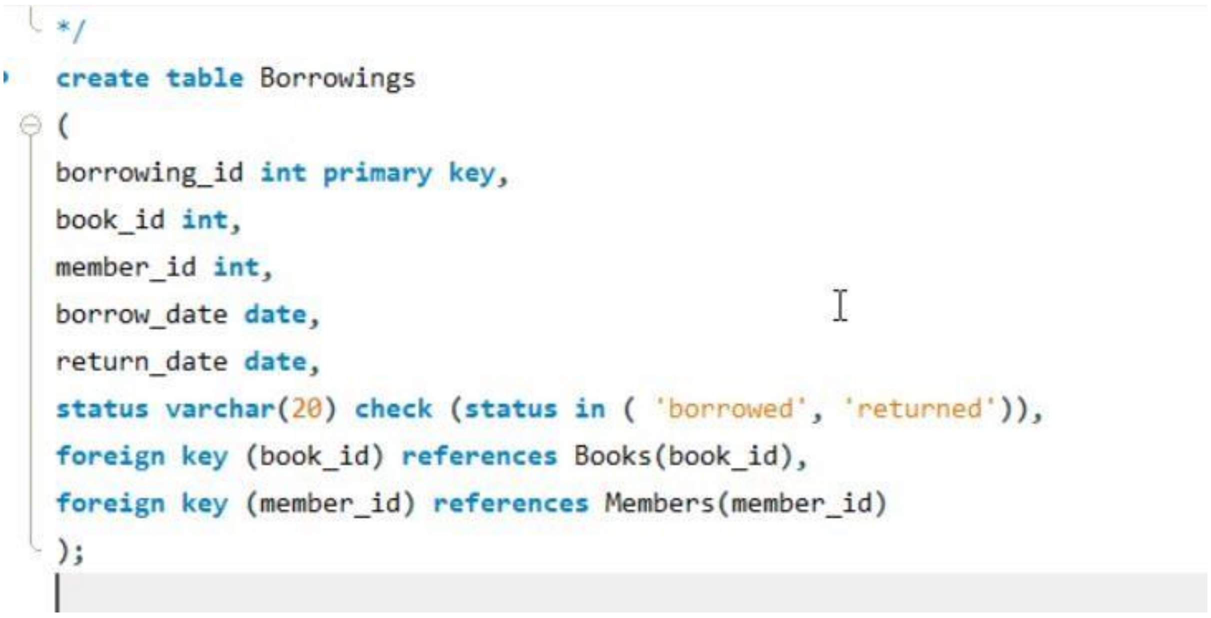
```
/**
Creating a memeber table
*/
create table Members
(
member_id int primary key,
member_name varchar(255) not null,
email varchar(255),
phone_number varchar(20)
);
desc Members;
```

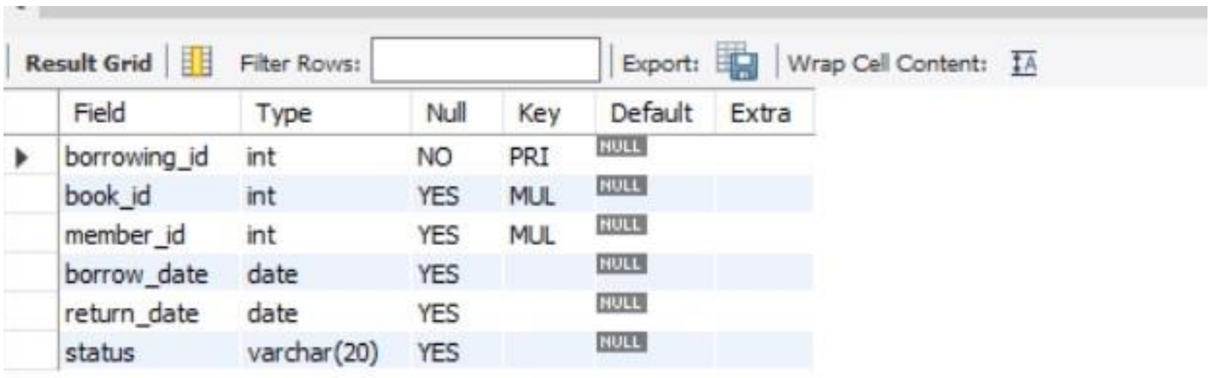

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	Field	Type	Null	Key	Default	Extra
▶	member_id	int	NO	PRI	NULL	
	member_name	varchar(255)	NO		NULL	
	email	varchar(255)	YES		NULL	
	phone_number	varchar(20)	YES		NULL	

4. **Borrowings:** This table will track the borrowing history of books by members.

- **Fields**: borrowing_id (Primary Key), book_id (Foreign Key referencing Books), member_id (Foreign Key referencing Members), borrow_date, return_date, status (e.g., 'borrowed', 'returned').


```
*/
create table Borrowings
(
borrowing_id int primary key,
book_id int,
member_id int,
borrow_date date,
return_date date,
status varchar(20) check (status in ( 'borrowed', 'returned')),
foreign key (book_id) references Books(book_id),
foreign key (member_id) references Members(member_id)
);
```



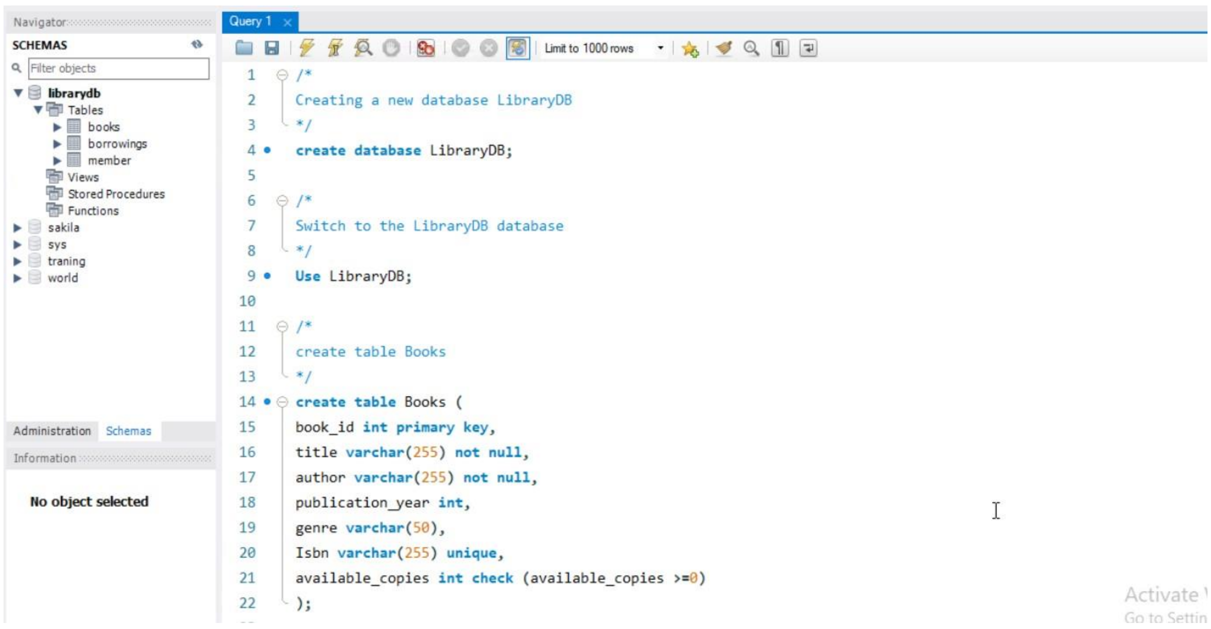
Field	Type	Null	Key	Default	Extra
borrowing_id	int	NO	PRI	NULL	
book_id	int	YES	MUL	NULL	
member_id	int	YES	MUL	NULL	
borrow_date	date	YES		NULL	
return_date	date	YES		NULL	
status	varchar(20)	YES		NULL	

In this schema:

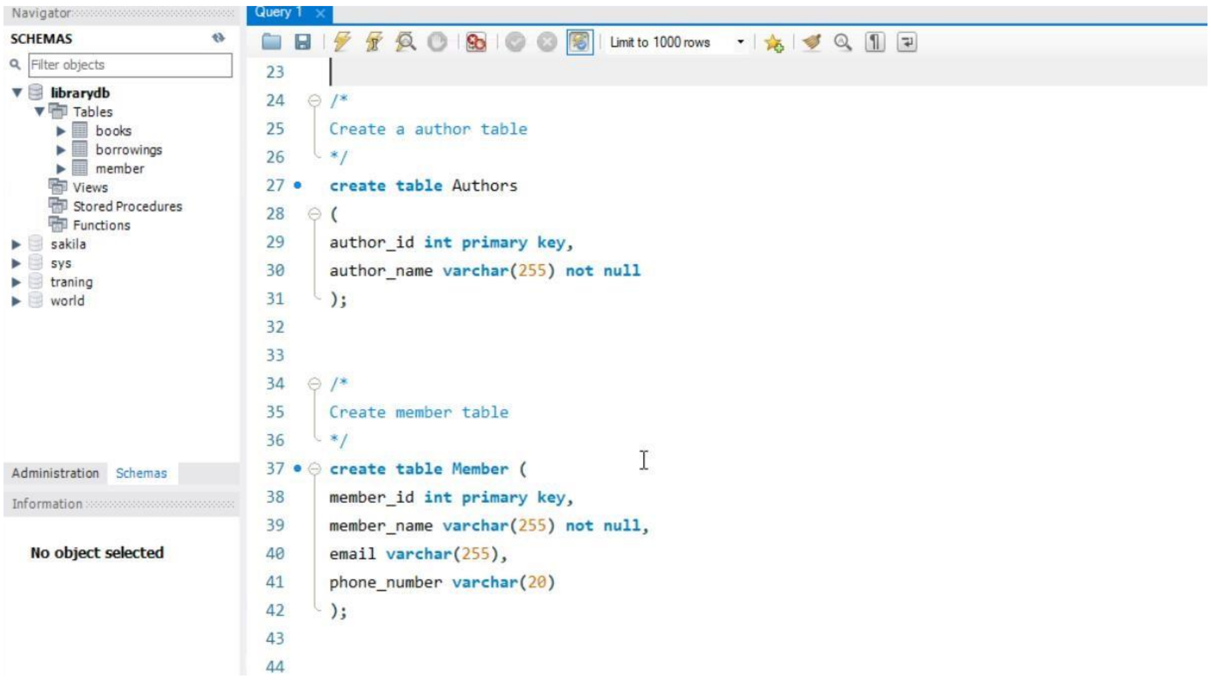
- The `Books` table has a primary key `book_id` and a unique constraint on `ISBN` to ensure each book has a unique identifier.
- The `Authors` table has a primary key `author_id` to uniquely identify authors.
- The `Members` table has a primary key `member_id` to uniquely identify members.
- The `Borrowings` table has a primary key `borrowing_id` and foreign keys `book_id` and `member_id` to establish relationships with the `Books` and `Members` tables, respectively. The `status` field ensures that only valid statuses ('borrowed' or 'returned') can be inserted.

ASSIGNMENT-3

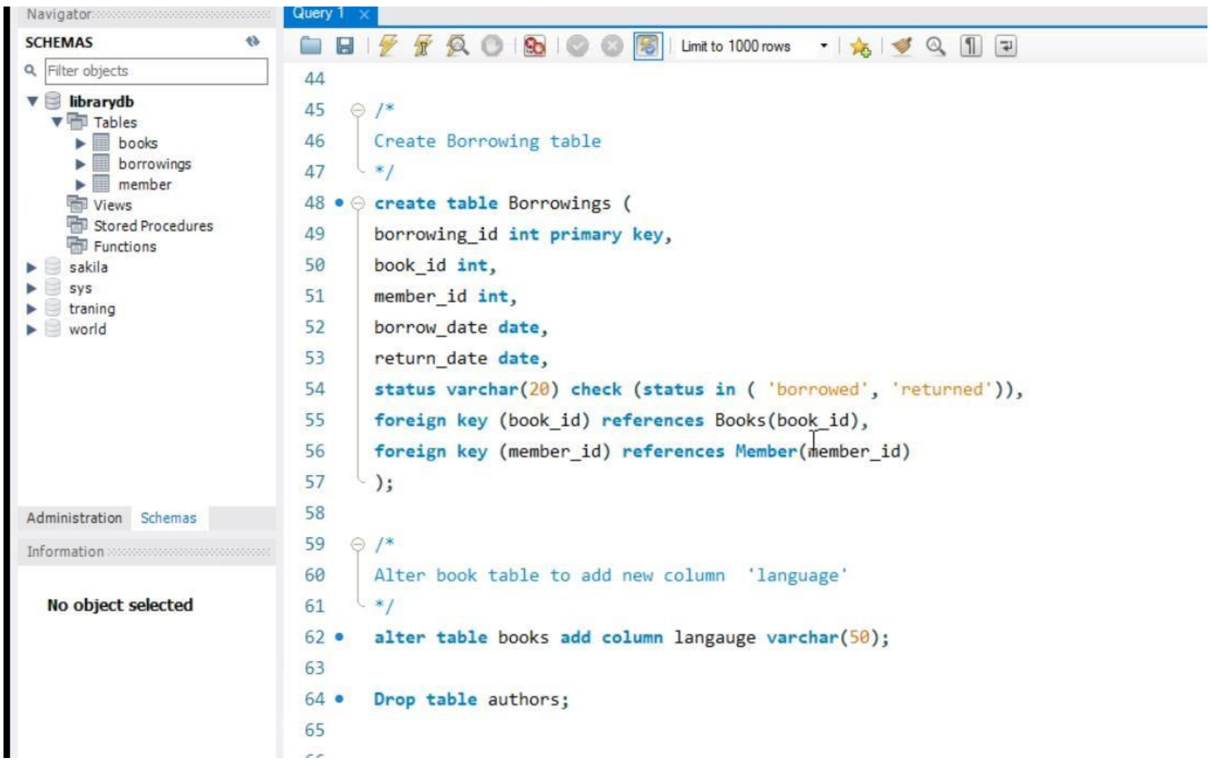
Write SQL statements to CREATE a new database and tables that reflect the library schema you designed earlier. Use ALTER statements to modify the table structures and DROP statements to remove a redundant table.



```
/*
Creating a new database LibraryDB
*/
create database LibraryDB;

/*
Switch to the LibraryDB database
*/
Use LibraryDB;

/*
create table Books
*/
create table Books (
book_id int primary key,
title varchar(255) not null,
author varchar(255) not null,
publication_year int,
genre varchar(50),
Isbn varchar(255) unique,
available_copies int check (available_copies >=0)
);
```



```sql

/*
Create a author table
*/
create table Authors
(
author_id int primary key,
author_name varchar(255) not null
);


/*
Create member table
*/
create table Member (
member_id int primary key,
member_name varchar(255) not null,
email varchar(255),
phone_number varchar(20)
);


```



```sql

/*
Create Borrowing table
*/
create table Borrowings (
borrowing_id int primary key,
book_id int,
member_id int,
borrow_date date,
return_date date,
status varchar(20) check (status in ( 'borrowed', 'returned')),
foreign key (book_id) references Books(book_id),
foreign key (member_id) references Member(member_id)
);

/*
Alter book table to add new column  'language'
*/
alter table books add column langauge varchar(50);

Drop table authors;

```

In this SQL script:

- We first create a new database called `LibraryDB` using `CREATE DATABASE` and then switch to it using `USE LibraryDB`.

- We create the `Books`, `Authors`, `Members`, and `Borrowings` tables based on the schema designed earlier.

- We use the `ALTER TABLE` statement to add a new column `language` to the `Books` table.

Finally, we drop the redundant `Authors` table using `DROP TABLE Authors;`. Note that this assumes the `Authors` table is redundant for this specific library system scenario